In [258]:

```python
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import sklearn
```

In [259]:

```python
car_sales_missing = pd.read_csv('car-sales-extended-missing-data.csv')
car_sales_missing.head()
```

Out[259]:

	Make	Colour	Odometer (KM)	Doors	Price
0	Honda	White	35431.0	4.0	15323.0
1	BMW	Blue	192714.0	5.0	19943.0
2	Honda	White	84714.0	4.0	28343.0
3	Toyota	White	154365.0	4.0	13434.0
4	Nissan	Blue	181577.0	3.0	14043.0

In [260]:

```python
# to check for missing data
car_sales_missing.isna().sum()
```

Out[260]:

```
Make             49
Colour           50
Odometer (KM)    50
Doors            50
Price            50
dtype: int64
```

In [261]:

```python
car_sales_missing.dropna(subset = ['Price'], inplace = True)
car_sales_missing
```

Out[261]:

	Make	Colour	Odometer (KM)	Doors	Price
0	Honda	White	35431.0	4.0	15323.0
1	BMW	Blue	192714.0	5.0	19943.0
2	Honda	White	84714.0	4.0	28343.0
3	Toyota	White	154365.0	4.0	13434.0
4	Nissan	Blue	181577.0	3.0	14043.0
...	...	...	...	...	...
995	Toyota	Black	35820.0	4.0	32042.0
996	NaN	White	155144.0	3.0	5716.0
997	Nissan	Blue	66604.0	4.0	31570.0
998	Honda	White	215883.0	4.0	4001.0
999	Toyota	Blue	248360.0	4.0	12732.0

950 rows × 5 columns

In [274]:

```python
# split into x and y
x = car_sales_missing.drop('Price', axis = 1)
y = car_sales_missing['Price']
```

In [275]:

```python
# to fix missing data/values with sklearn

from sklearn.impute import SimpleImputer
from sklearn.compose import ColumnTransformer

# fill categorical values 'missing' & numerical values with 'mean'
cat_imputer = SimpleImputer(strategy = 'constant', fill_value = 'missing')
door_imputer = SimpleImputer(strategy = 'median')
num_imputer = SimpleImputer(strategy = 'median')

# Define colums
cat_features = ['Make', 'Colour']
door_feature = ['Doors']
num_features = ['Odometer (KM)']

# create an inputer (sosmething that fills missing data)
imputer = ColumnTransformer([
    ('cat_imputer', cat_imputer, cat_features),
    ('door_imputer', door_imputer, door_feature),
    ('num_imputer', num_imputer, num_features)
])

# transform the data
filled_x = imputer.fit_transform(x)
filled_x

```

Out[275]:

```
array([['Honda', 'White', 4.0, 35431.0],
       ['BMW', 'Blue', 5.0, 192714.0],
       ['Honda', 'White', 4.0, 84714.0],
       ...,
       ['Nissan', 'Blue', 4.0, 66604.0],
       ['Honda', 'White', 4.0, 215883.0],
       ['Toyota', 'Blue', 4.0, 248360.0]], dtype=object)
```

In [276]:

```python
car_sales_filled = pd.DataFrame(filled_x,
                               columns = ['Make', 'Colour', 'Doors', 'Odometer (KM)'])
car_sales_filled.head()
```

Out[276]:

	Make	Colour	Doors	Odometer (KM)
0	Honda	White	4.0	35431.0
1	BMW	Blue	5.0	192714.0
2	Honda	White	4.0	84714.0
3	Toyota	White	4.0	154365.0
4	Nissan	Blue	3.0	181577.0

In [277]:

```python
car_sales_filled.isna().sum()
```

Out[277]:

```
Make             0
Colour           0
Doors            0
Odometer (KM)    0
dtype: int64
```

In [278]:

```python
car_sales_filled.head()
```

Out[278]:

	Make	Colour	Doors	Odometer (KM)
0	Honda	White	4.0	35431.0
1	BMW	Blue	5.0	192714.0
2	Honda	White	4.0	84714.0
3	Toyota	White	4.0	154365.0
4	Nissan	Blue	3.0	181577.0

In [279]:

```
# let's try and convert our data to numbers
from sklearn.preprocessing import OneHotEncoder
from sklearn.compose import ColumnTransformer

# find the category features to convert to numbers
categorical_features = ['Make', 'Colour', 'Doors']
one_hot = OneHotEncoder()
transformer = ColumnTransformer([('one_hot',
                                  one_hot,
                                  categorical_features)],
                                remainder = 'passthrough')

transformed_x = transformer.fit_transform(car_sales_filled)
transformed_x
```

Out[279]:

```
<950x15 sparse matrix of type '<class 'numpy.float64'>'
        with 3800 stored elements in Compressed Sparse Row format>
```

In [280]:

```
# Now we've got our data as numbers and filled (no missing values)
# let's fit a model
np.random.seed(42)
from sklearn.ensemble import RandomForestRegressor
from sklearn.model_selection import train_test_split

x_train, x_test, y_train, y_test = train_test_split(transformed_x,
                                                    y,
                                                     test_size = 0.2)
model = RandomForestRegressor()
model.fit(x_train, y_train)
model.score(x_test, y_test)

```

Out[280]:

```
0.22034702153671681
```

In [282]:

```
model.predict(x_test)
```

```
Out[282]: array([17192.59      , 20654.13      , 12414.55      ,  9436.56      ,
                 11165.25      , 11178.02      , 15664.46      , 10307.4       ,
                 17105.        , 15316.12995349,  8361.02      , 14365.        ,
                  8493.04      , 10081.6       , 13889.59      , 19708.52      ,
                 15009.24      ,  7506.11      , 11269.28      , 14660.69      ,
                 11147.11      , 17882.96916667, 20306.45      , 27367.26      ,
                  9084.12      , 21191.73      , 12919.59      ,  8007.88      ,
                 20345.32      , 17089.77      , 11372.4       , 17132.11      ,
                 10601.83      , 11266.11083333, 27719.93      , 15011.21      ,
                 12272.11      , 13579.73      , 21812.46      ,  9500.7       ,
                 15673.27      , 20796.25      , 25363.61      , 15537.17      ,
                 13998.01057143, 11624.16      , 14903.47      ,  8387.02      ,
                 15224.81      , 13027.37      , 11271.06      , 21285.78      ,
                 14986.42      ,  5720.92      , 11703.23      ,  9569.59      ,
                 14879.77      , 11623.94      , 10607.7785    , 15025.04      ,
                 13719.8       , 22603.23075216, 11529.38      , 17591.58      ,
                 24031.29      , 26636.14      , 11424.85      , 14331.98      ,
                 18421.83      , 18828.26      , 13687.52      , 20846.62      ,
                 20528.29      , 15256.82      , 11394.66      ,  7376.84      ,
                  9070.63      , 11269.03630159, 29173.        , 11762.7       ,
                 16496.64      , 13555.12      , 12173.09833333, 13572.91      ,
                  8706.77      , 19495.35      ,  7752.52      , 20144.75      ,
                 16365.68      , 11372.4       , 13105.47420635,  7281.62      ,
                 10106.73      , 10348.56      , 19139.26      ,  6174.5       ,
                 15316.12995349, 11876.78      ,  7780.42      , 26035.04      ,
                 15640.22      , 12183.18      , 25786.0593254 , 18637.        ,
                 38736.12      , 11575.24      ,  8384.3       , 11283.39533333,
                 15368.93      , 17055.43      , 24461.4265    ,  6583.11      ,
                  7352.24      ,  8593.22      , 19010.02672619, 11686.01344444,
                  5570.08      , 20700.84      , 12148.31      , 29006.79      ,
                 22232.56      , 11834.14      , 26293.78      , 15403.23      ,
                 13328.36      , 15298.23      , 17766.66      , 12404.69      ,
                 14626.75      , 19671.78      , 22244.48      , 12561.72      ,
                 19546.2       , 24574.75      , 13228.97      ,  8830.19      ,
                 16404.28      , 12223.1       , 27163.45      , 17729.97      ,
                 11927.8       , 16456.72      , 26050.32      , 35928.41      ,
                 16569.89      , 12761.83      , 15230.04666667,  9630.45      ,
                 19364.59      , 15823.19      , 22058.61      , 19507.47      ,
                 18251.4       ,  7546.58      , 14803.65      , 19899.58      ,
                 10607.7785    , 10228.49      , 12264.09      , 20107.1       ,
                 17207.67      , 15316.12995349, 14662.        , 10667.32      ,
                 13581.84      , 23251.21      , 18357.13      , 12659.204     ,
                 18205.8       , 11716.11344444, 22329.31      , 12673.46      ,
       19120.06      , 12289.17      , 11819.76      , 18639.89      ,
       22449.84      , 21956.29      , 47179.93      , 12141.27      ,
       11446.24      , 12035.81      , 38010.29      , 18099.22      ,
        8885.73      , 12920.09      , 10226.72      , 12353.33      ,
       23643.34      , 19729.1       ])
```

```
In [ ]:
```